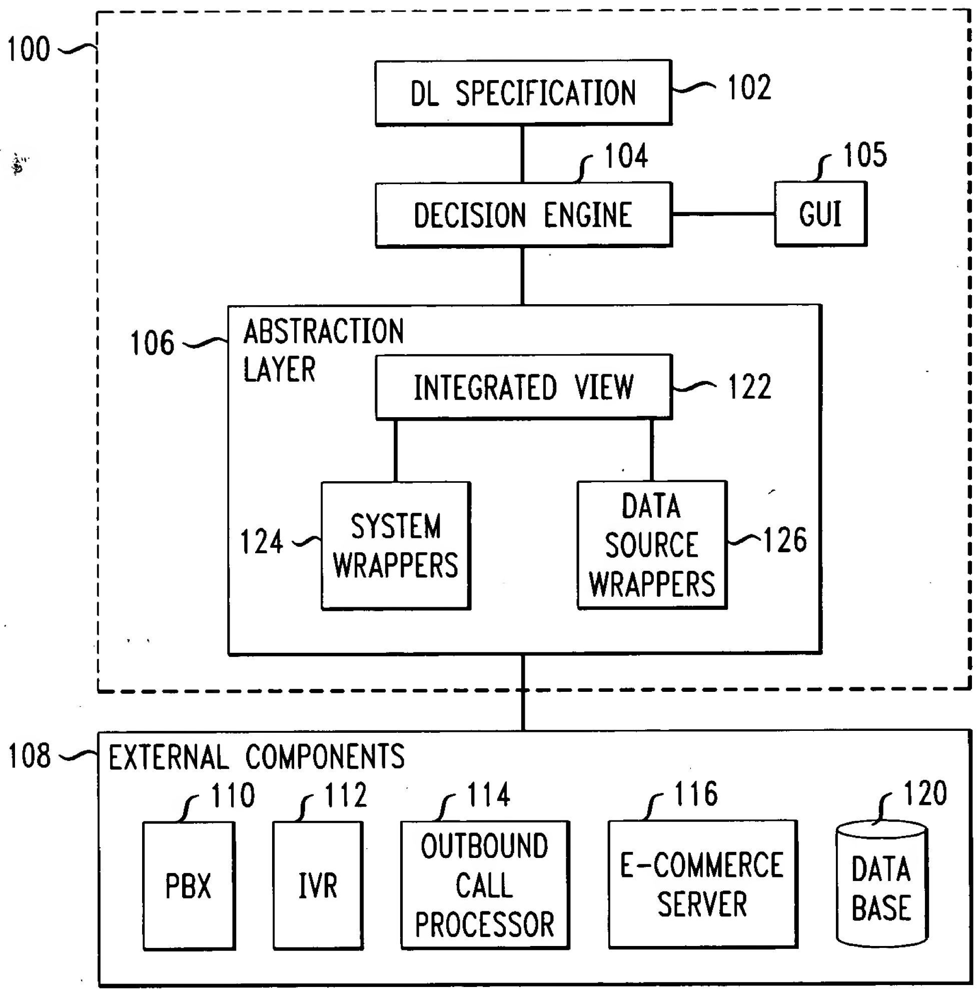
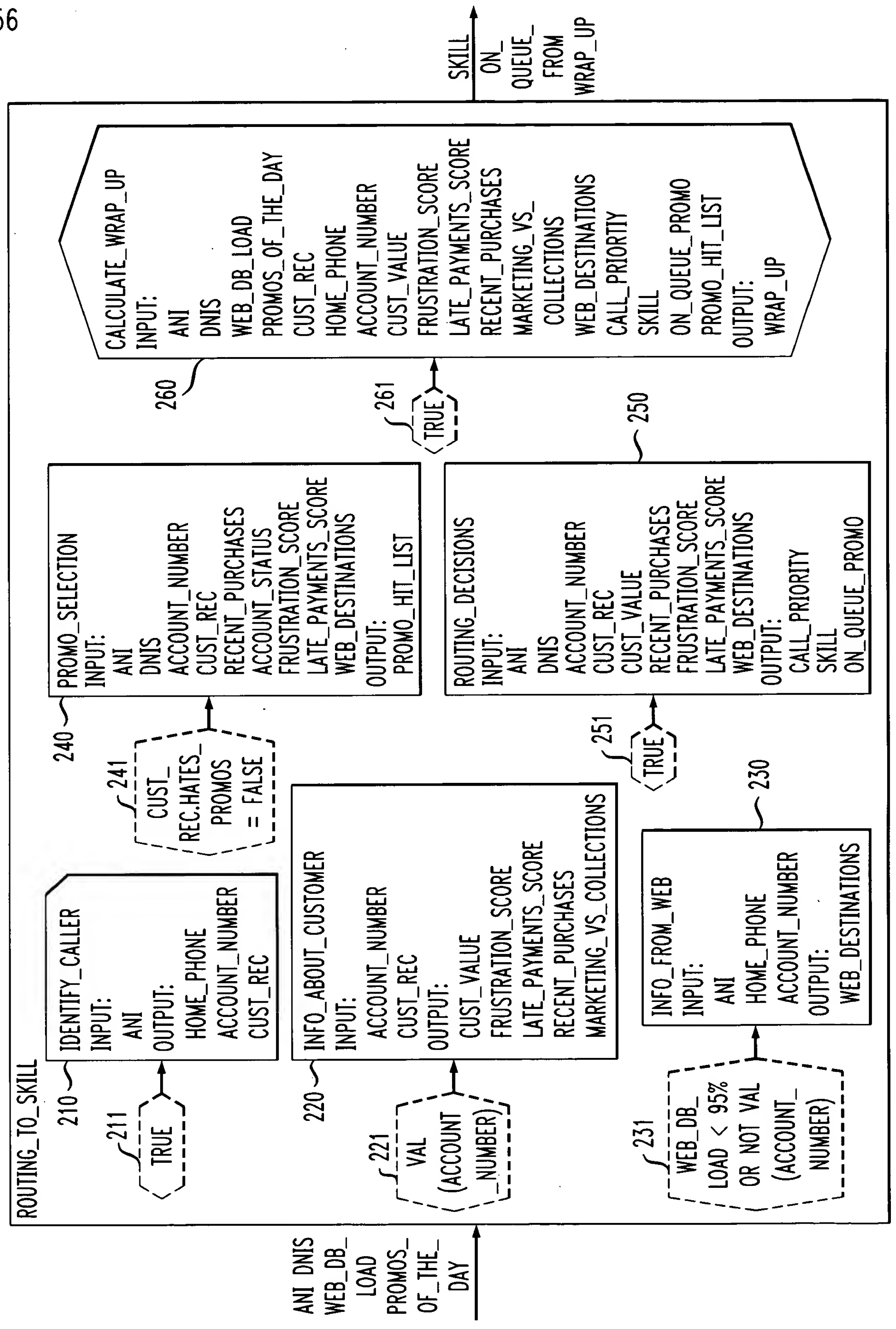
09/251998

FIG. 1



09/251998

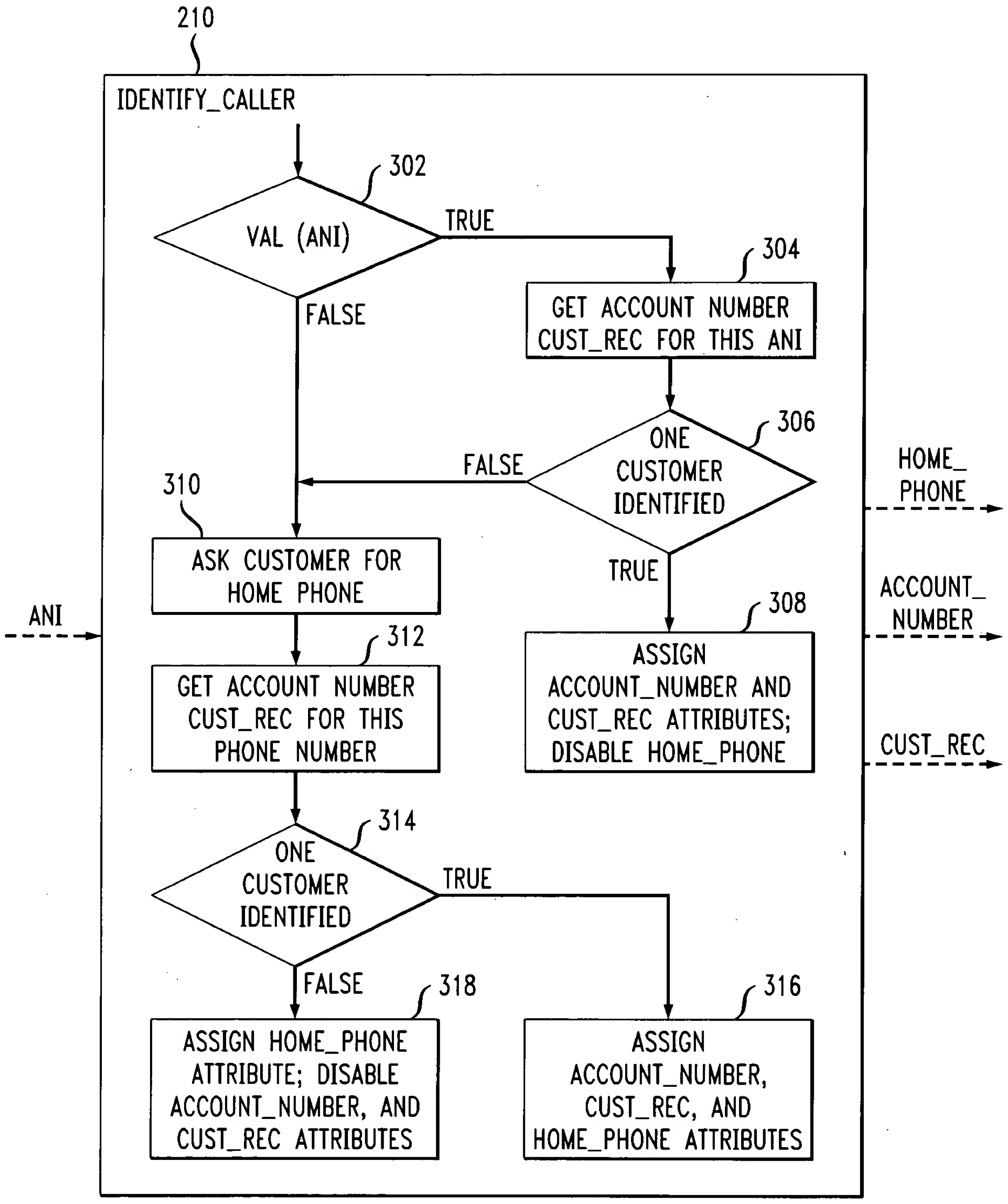
FIG. 2


ANI DNIS WEB_DB_ LOAD PROMOS_ OF_THE_ DAY
ROUTING_TO_SKILL
211 TRUE
210
IDENTIFY_CALLER INPUT: ANI OUTPUT: HOME_PHONE ACCOUNT_NUMBER CUST_REC
221 VAL (ACCOUNT _NUMBER)
220
INFO_ABOUT_CUSTOMER INPUT: ACCOUNT_NUMBER CUST_REC OUTPUT: CUST_VALUE FRUSTRATION_SCORE LATE_PAYMENTS_SCORE RECENT_PURCHASES MARKETING_VS_COLLECTIONS
231 WEB_DB_ LOAD < 95% OR NOT VAL (ACCOUNT_ NUMBER)
230
INFO_FROM_WEB INPUT: ANI HOME_PHONE ACCOUNT_NUMBER OUTPUT: WEB_DESTINATIONS
241 CUST_ REC.HATES_ PROMOS = FALSE
240
PROMO_SELECTION INPUT: ANI DNIS ACCOUNT_NUMBER CUST_REC RECENT_PURCHASES ACCOUNT_STATUS FRUSTRATION_SCORE LATE_PAYMENTS_SCORE WEB_DESTINATIONS OUTPUT: PROMO_HIT_LIST
251 TRUE
250
ROUTING_DECISIONS INPUT: ANI DNIS ACCOUNT_NUMBER CUST_REC CUST_VALUE RECENT_PURCHASES FRUSTRATION_SCORE LATE_PAYMENTS_SCORE WEB_DESTINATIONS OUTPUT: CALL_PRIORITY SKILL ON_QUEUE_PROMO
261 TRUE
260
CALCULATE_WRAP_UP INPUT: ANI DNIS WEB_DB_LOAD PROMOS_OF_THE_DAY CUST_REC HOME_PHONE ACCOUNT_NUMBER CUST_VALUE FRUSTRATION_SCORE LATE_PAYMENTS_SCORE RECENT_PURCHASES MARKETING_VS_ COLLECTIONS WEB_DESTINATIONS CALL_PRIORTIY SKILL ON_QUEUE_PROMO PROMO_HIT_LIST OUTPUT: WRAP_UP
SKILL ON_ QUEUE_ FROM WRAP_UP

JAN 2 7 2003

09/251998

FIG. 3


210
IDENTIFY_CALLER
302
VAL (ANI)
TRUE
FALSE
304
GET ACCOUNT NUMBER CUST_REC FOR THIS ANI
306
ONE CUSTOMER IDENTIFIED
FALSE
TRUE
310
ASK CUSTOMER FOR HOME PHONE
312
GET ACCOUNT NUMBER CUST_REC FOR THIS PHONE NUMBER
308
ASSIGN ACCOUNT_NUMBER AND CUST_REC ATTRIBUTES; DISABLE HOME_PHONE
314
ONE CUSTOMER IDENTIFIED
TRUE
FALSE
318
ASSIGN HOME_PHONE ATTRIBUTE; DISABLE ACCOUNT_NUMBER, AND CUST_REC ATTRIBUTES
316
ASSIGN ACCOUNT_NUMBER, CUST_REC, AND HOME_PHONE ATTRIBUTES
ANI
HOME_PHONE
ACCOUNT_NUMBER
CUST_REC

09/251998

FIG. 4

```
Module:   identify_caller
    Submodule of:   routing_to_skill
    Input attributes:      ANI : 9digits
    Output attributes:     home_phone : 9digits
                           account_number : 15digits
                           cust_rec : tuple (name: string,
                                             address: string,
                                             card_color: ("platinum",
                                             "gold", "green"),
                                             hates_promos? : boolean,
                                             estimated_income_bracket :
                                             ("0-10K", ">10K-20K",...,
                                             ">100K-150K", ">150"),
                                             last_sent_bonus_check:date)
    Enabling condition:    true
    Type:               flowchart
    Computation:        See Fig. 3
    Side-effect:        yes
    Side Effect function:  (IVR Dip)
```

09/251998

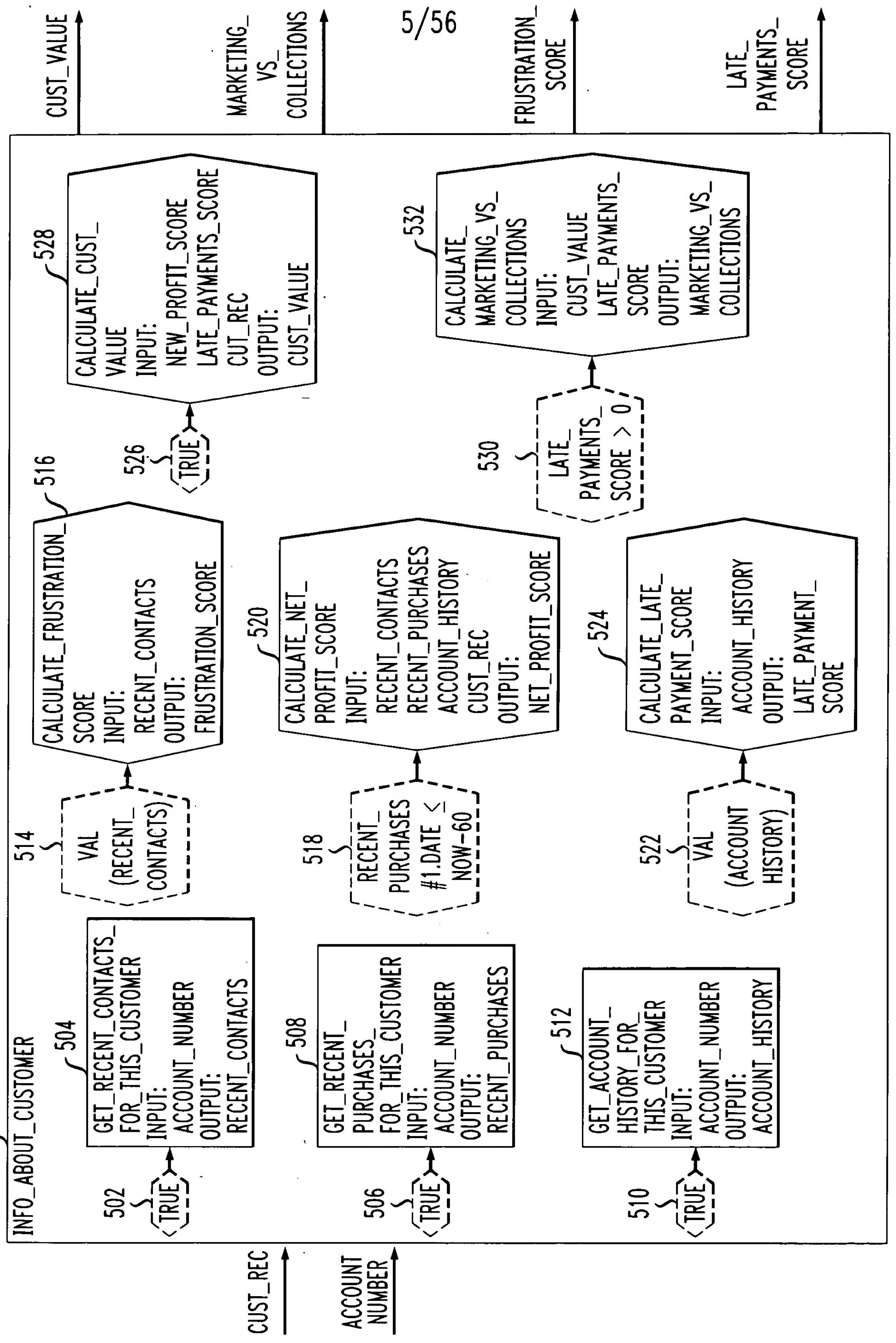

INFO_ABOUT_CUSTOMER
220
CUST_REC
ACCOUNT NUMBER
502
TRUE
504
GET_RECENT_CONTACTS_FOR_THIS_CUSTOMER INPUT: ACCOUNT_NUMBER OUTPUT: RECENT_CONTACTS
506
TRUE
508
GET_RECENT_PURCHASES_FOR_THIS_CUSTOMER INPUT: ACCOUNT_NUMBER OUTPUT: RECENT_PURCHASES
510
TRUE
512
GET_ACCOUNT_HISTORY_FOR_THIS_CUSTOMER INPUT: ACCOUNT_NUMBER OUTPUT: ACCOUNT_HISTORY
514
VAL (RECENT_CONTACTS)
516
CALCULATE_FRUSTRATION_SCORE INPUT: RECENT_CONTACTS OUTPUT: FRUSTRATION_SCORE
518
RECENT_PURCHASES #1.DATE ≤ NOW-60
520
CALCULATE_NET_PROFIT_SCORE INPUT: RECENT_CONTACTS RECENT_PURCHASES ACCOUNT_HISTORY CUST_REC OUTPUT: NET_PROFIT_SCORE
522
VAL (ACCOUNT HISTORY)
524
CALCULATE_LATE_PAYMENT_SCORE INPUT: ACCOUNT_HISTORY OUTPUT: LATE_PAYMENT_SCORE
526
TRUE
528
CALCULATE_CUST_VALUE INPUT: NEW_PROFIT_SCORE LATE_PAYMENTS_SCORE CUT_REC OUTPUT: CUST_VALUE
530
LATE_PAYMENTS_SCORE > 0
532
CALCULATE_MARKETING_VS_COLLECTIONS INPUT: CUST_VALUE LATE_PAYMENTS_SCORE OUTPUT: MARKETING_VS_COLLECTIONS
CUST_VALUE
MARKETING_VS_COLLECTIONS
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE

FIG. 5

09/251998

FIG. 6

```
Module:    info_about_customer
    Submodule of:   routing_to_skill
    Input attributes:       account_number
                            cust_rec

    Output attributes:      cust_value : [1..10]
                            frustration_score : [1..10]
                            late_payments_score : [1..10]
                            recent_purchases :list(tuple( date : date,
                                                          item : 20digit,
                                                          qty : int,
                                                          amount: $value ))
                            marketing_vs_collections : {"market",
                            "collect"}

Enabling condition:         VAL(account_number)
    Type:               declarative
    Side-effect:      no
```

FIG. 7

```
Module:    info_from_web
    Submodule of:   routing_to_skill
    Input attributes:       ANI
                            home_phone
                            account_number

    Output attributes:      web_destinations : list(tuple(regions: set of
                                                 {"Australia","Asia",...
                                                 "NAmerica-US", "US"},
                                               itinerary:web_form_content,
                                               date_last_modified : date ))

    Enabling condition:     web_db_load < 95% or not VAL(account_number)

    Type:                   foreign

    Computation:            get_web_data(ANI, home_phone, account_number)

    Side-effect:            no
```

FIG. 8

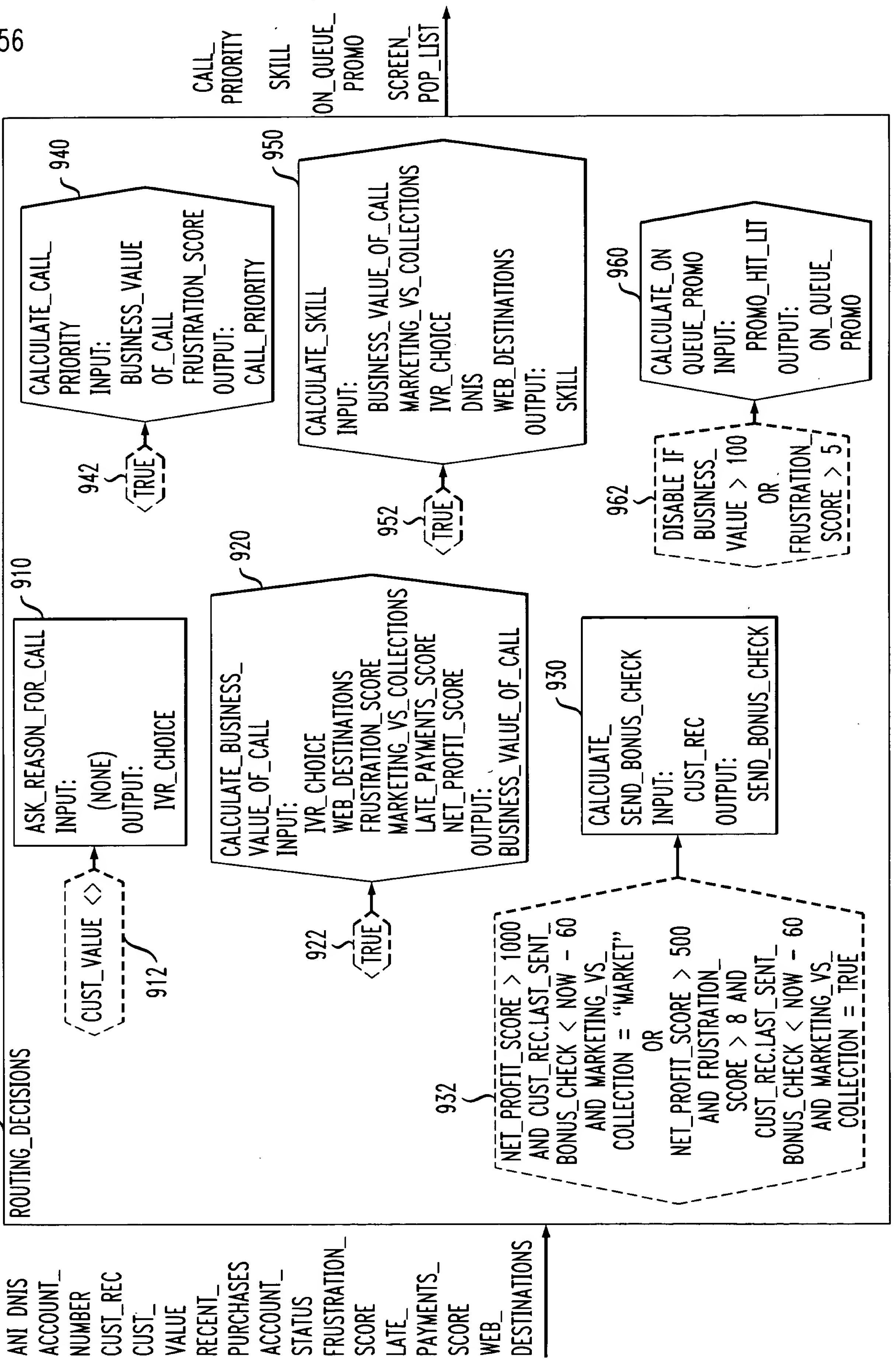
```
Module:    promo_selection

     Submodule of:   routing_to_skill

     Input attributes:        ANI
                              DNIS
                              account_number
                              cust_rec
                              cust_value
                              recent_purchases
                              frustration_score
                              late_payments_score
                              web_destinations

     Output attributes:       promo_hit_list : list ( promo_message )

     Enabling condition:      cust_rec.hates_promos? = false

     Type:                    foreign

     Computation:             get_promo_hit_list(ANI, DNIS, account_number,
                              cust_rec, cust_value, recent_purchases,
                              account_status, frustration_score,
                              late_payments_score, web_destinations)
     Side-effect:             no
```

09/251998

Hull 5-4-1-4
Serial No.: 09/251,998
Ryan, Mason & Lewis, LLP; R. J. Mauri (203) 255-6560

09/251998

FIG. 9


ANI DNIS
ACCOUNT_NUMBER
CUST_REC
CUST_VALUE
RECENT_PURCHASES
ACCOUNT_STATUS
FRUSTRATION_SCORE
LATE_PAYMENTS_SCORE
WEB_DESTINATIONS
230
ROUTING_DECISIONS
910
ASK_REASON_FOR_CALL INPUT: (NONE) OUTPUT: IVR_CHOICE
912
CUST_VALUE <>
920
CALCULATE_BUSINESS_VALUE_OF_CALL INPUT: IVR_CHOICE WEB_DESTINATIONS FRUSTRATION_SCORE MARKETING_VS_COLLECTIONS LATE_PAYMENTS_SCORE NET_PROFIT_SCORE OUTPUT: BUSINESS_VALUE_OF_CALL
922
TRUE
930
CALCULATE_SEND_BONUS_CHECK INPUT: CUST_REC OUTPUT: SEND_BONUS_CHECK
932
NET_PROFIT_SCORE > 1000 AND CUST_REC.LAST_SENT_BONUS_CHECK < NOW - 60 AND MARKETING_VS_COLLECTION = "MARKET" OR NET_PROFIT_SCORE > 500 AND FRUSTRATION_SCORE > 8 AND CUST_REC.LAST_SENT_BONUS_CHECK < NOW - 60 AND MARKETING_VS_COLLECTION = TRUE
940
CALCULATE_CALL_PRIORITY INPUT: BUSINESS_VALUE OF_CALL FRUSTRATION_SCORE OUTPUT: CALL_PRIORITY
942
TRUE
950
CALCULATE_SKILL INPUT: BUSINESS_VALUE_OF_CALL MARKETING_VS_COLLECTIONS IVR_CHOICE DNIS WEB_DESTINATIONS OUTPUT: SKILL
952
TRUE
960
CALCULATE_ON QUEUE_PROMO INPUT: PROMO_HIT_LIT OUTPUT: ON_QUEUE_PROMO
962
DISABLE IF BUSINESS_VALUE > 100 OR FRUSTRATION_SCORE > 5
CALL_PRIORITY
SKILL
ON_QUEUE_PROMO
SCREEN_POP_LIST

09/251998

FIG. 10

```
 1 Module:    routing_decisions
 2     Submodule of:  routing_to_skill
 3     Input attributes:      ANI
 4                            DNIS
 5                            account_number
 6                            cust_rec
 7                            cust_value
 8                            recent_purchases
 9                            frustration_score
10                            late_payments_score
11                            web_destinations
12     Output attributes:     call_priority : [1..4] \\corresponds to "low",
13                                   "med", "high", "top"
14                            skill : {"norm_tier_dom", "norm_tier_intl",
15                                   "australia_promo", "high_tier",
16                                          collections"}
17                            on_queue_promo : message_identifier
18                            screen_pop_list : list ( screen_entry )
19     Enabling condition:    true
20     Type:                  declarative
21     Side-effect:           yes
```

FIG. 11

```
Module:   calculate_wrap_up

     Submodule of:    routing_to_skill

     Input attributes:     Ani
                           dnis
                           Web_DB_Load
                           Promos_Of_The_Day
                           Cust_Rec
                           Home_Phone
                           Account_Number
                           Cust_Value
                           Frustration_Score
                           Late_Payments_Score
                           Recent_Purchases
                           Marketing_VS_Collections
                           Web_Destinations
                           Call_Priority
                           Skill
                           On_Queue_Promo
                           Screen_Pop_List
                           Promo_Hit_List

     Output attributes:    wrap_up : set ( tuple ( att_name: string,
                                                   value: string ))
     Enabling condition:   true

     Type:            decision

     Computation:
        Rules:             if true then wrap_up <- (att_name: "DNIS",
                                         value : convert-to-string (DNIS))
                           if true then wrap_up <- (att_name: "ANI",
                                         value: convert-to-string (ANI))
                           if true then wrap_up <- (att_name: "skill",
                                         value: skill)
                           if web_destinations not empty then wrap_up <-
                                        (att_name: \"web_destinations",
                                         value: (convert-to-string
                                                (web_destinations))
                           if cust_rec.card_color = "gold" <-
                                        (att_name:"frustration_score",
                                         value: convert-to-string
                                         (frustration_score))
     Combining policy:         wrap_up_cp //use contributions of all
                                            rules with true condition
     Side-effect:              yes

     Side-effect function:     write_into_archive ( wrap_up )
```

09/251998

FIG. 12

```
Module:    get_recent_contacts_for_this_customer

    Submodule of:   info_about_customer

    Input attributes:     account_number

    Output attributes:    recent_contacts : list ( tuple ( date: date,
                                                   event: event_type,
                                                   delay_during_contact: int,
                                                       \\ minutes
                                                   delay_before_shipment: int
                                                       \\ days
                                                   amount: $value ) )
    Enabling condition:   true

    Type:                 foreign

    Computation:          using recent_contacts_db
                          select date,event,amount
                          from contact_db
                          where acct_num = account_number

    Side-effect:          no
```

09/251998

FIG. 13

```
Module:    get_recent_purchases_for_this_customer

    Submodule of:    info_about_customer

    Input attributes:      account_number

    Output attributes:     recent_purchases : list ( tuple ( date: date,
                                              item : 20digit,
                                              qty : int,
                                              amount : $value ) )
    Enabling condition:    true

    Type:            foreign

    Computation:           using purchase_db
                           select date,item,qty,amount
                           from purchases
                           where acct_num = account_number

    Side-effect:           no
```

09/251998

FIG. 14

```
Module:    get_account_history_for_this_customer

    Submodule of:    info_about_customer

    Input attributes:     account_number

    Output attributes:   account_history : tuple ( overdue amount:
                                                        $value,
                                                 number_days_overdue:
                                                        int,
                                                 history: list ( tuple (
                                                        date: date,
                                                 item : 20digit,
                                                 amount : $value ) ) )
    Enabling condition:  true

    Type:            foreign

    Computation:          using account_history_db
                          select over_amt, num_days,history
                          from account_history
                          where acct_num = account_number

    Side-effect:          no
```

09/251998

FIG. 15

```
Module:  calculate_frustration_score

    Submodule of:  info_about_customer

    Input attributes:     recent_contacts

    Output attributes:    frustration_score : [1..10]

    Enabling condition:   VAL(recent_contacts)

    Type:            decision

    Computation:
        Rules:            if recent_contacts#1 defined then
                          frustration_score <-
                                    (value/50) *
                                    [(delay_during_contact/2) +
                                    max(0,delay_before_shipment -
                                    10)/3]

                          if recent_contacts#2 defined then
                          frustration_score <-
                                    (value/100) *
                                    [(delay_during_contact/2) +
                                    max (0,delay_before_shipment -
                                    10)/3]


        Combining policy: frustration_score_cp //add contributions
                                                 of true rules and
                                                 round up, to max
                                                 of 10

    Side-effect:          no
```

09/251998

FIG. 16

```
Module:   calculate_net_profit_score
  Submodule of:   info_about_customer
  Input attributes:     recent_contacts,
                        recent_purchases,
                        account_history,
                        cust_rec
  Output attributes:    net_profit_score
  Enabling condition:   recent_purchases#1.date<=now-60
  Type:           decision
  Computation:
      Rules:                        if recent_purchases not empty then
                                    net_profit_score <-
                                    10% * sum (recent_purchases#i.amount
                                    where recent_purchases#i.date > now -
                                    60)
                                    if recent_contacts not empty then
                                    net_profit_score <-
                                    -( 5 * count ( recent_contacts#i
                                    where recent_contacts#i.type =
                                    "complaint"))

                                    if account_history.overdue_amount > 0
                                    then net_profit_score <-
                                    - account_history.overdue_amount *
                                    account_history.number_days_overdue / 30

                                    if cust_rec.card_color = "platinum" then
                                    net_profit_score <- 100

                                    if cust_rec.card_color = "gold" then
                                    net_profit_score <- 50

                                    if cust_rec.card_color = "green" then
                                    net_profit_score <- 10

                                    if DISABLED(cust_rec) then
                                    net_profit_score <- 20

       Combining policy:            net_profit_score_cp //add contributions
                                                  of rules with true
                                                  conditions

  Side-effect:          no
```

09/251998

FIG. 17

```
Module:   calculate_late_payment_score
  Submodule of:   info_about_customer
  Input attributes:      account_history
  Output attributes:     late_payment_score
  Enabling condition:    VAL(account_history)
  Type:           decision
  Computation:
      Rules:                      if cust_rec.card_color = "platinum" then
                                  late_payments_score <-
                                  (account_history.overdue_amount
                                  number_of_days_overdue)/100

                                  if cust_rec.card_color = "gold" then
                                  late_payments_score <-
                                  (account_history.overdue_amount *
                                  number_of days_overdue)/50

                                  if cust_rec.card_color = "green" then
                                  late_payments_score <-
                                  (account_history.overdue_amount *
                                  number_of days_overdue)/10

      Combining policy:           late_payment_score_cp //rule with true
                                                          condition wins;
                                                          default is 0

  Side-effect:         no
```

09/251998

FIG. 18

```
Module:    calculate_cust_value
   Submodule of:   info_about_customer
   Input attributes:      net_profit_score,
                          late_payments_score,
                          cust_rec

   Output attributes:     cust_value
   Enabling condition:    true
   Type:              decision
   Computation:
        Rules:            if VAL(net_profit_score) then cust_value <-
                                    (1 - 1/net_profit_score) * 75

                          if cust_rec.card_color = "platinum" then
                          cust_value <- 20

                          if cust_rec.card_color = "gold" then cust_value
                          <- 10

                          if cust_rec.card_color = "green" then
                          cust_value <- 5

                          if VAL (frustration_score) then cust_value ←
                          5*frustration_score

      Combining policy:   calculate_cust_val_cp //Add values of true
                                                 rules and round up, to
                                                 max of 100, default is
                                                 0

   Side-effect:           no
```

09/251998

09/251998

FIG. 19

```
Module:    calculate_marketing_vs_collections
  Submodule of:   info_about_customer
  Input attributes:       cust_value,
                          late_payments_score
  Output attributes:      marketing_vs_collections
  Enabling condition:     late_payments_score > 0
  Type:           decision
  Computation:
      Rules:              if late_payments_score > f(cust_value) then
                          marketing_vs_collections <- "collect"
                          // f is function from [1..100] into [1..10],
                          // it could be linear, i.e., f(cust_value) =
                          //   cust_value/10
                          // or it could be shallower in beginning and
                               steeper
                          // towards end


     Combining policy:  marketing_vs_collection_cp //default is
                                                     "marketing",
                                                     any rule
                                                     with true
                                                     condition
                                                     wins

  Side-effect:          no
```

FIG. 20

```
Module:  Ask_Reason_For_Call
  Submodule of:   routing_decisions
  Input attributes:      < none >
  Output attributes:    IVR_choice
  Enabling condition:   cust_value < 7 and DNIS not =
                        "Australia_promotion"
  Type:             foreign
  Computation:          x := IVR_dip( question(2) ) ;
                        if x = 1 then IVR_choice := "dom";
                        else if x = 2 the IVR_choice := "intl";
                        else IVR_choice[state] = EXC and
                        IVR_choice[EXC]=1

    Side-effect:          yes

    Side-effect-function: IVR_dip( question (2) )
```

09/251998

FIG. 21

```
Module:  calculate_business_value_of_call
    Submodule of:  routing_decisions
    Input attributes:     IVR_choice,
                          web_destinations,
                          frustration_score,
                          marketing_vs_collections,
                          late_payments_score,
                          net_profit_score
    Output attributes:    business_value_of_call : int
    Enabling condition:   true
    Type:            decision
    Computation:
        Rules:
                     if true then business_value_of_call <-
                          (cust_value/50 * net_profit_score)

                     if true then business_value_of_call <-
                          10*frustration_score

                     if DNIS = "Australia_promtion" then
                          business_value_of_call <- 100

                     if "Australia" in web_destinations[i].regions for
                          some i where
                          web_destinations[i].date_last_modified > now -
                          30
                          then business_value_of_call <- 100

                     if IVR_choice = "intl" then business_value_of_call <-
                          50

                     if marketing_vs_collections = "collect" then
                          business_value_of_call <-
                          (late_payments_score *
                          account_history.overdue_amount)/5

Combining policy:    business_value_of_call_cp // Add contributions of
                                                  rules with true
                                                  conditions and round up,
                                                  default is 0

    Side-effect:          no
```

09/251998

FIG. 22

```
Module:  Calculate_send_bonus_check

   Submodule of:   routing_decisions

   Input attributes:        cust_rec

   Output attributes:       send_bonus_check?

   Enabling condition:      if net_profit_score > 1000
                            and cust_rec.last_sent_bonus_check < now - 60
                            and marketing_vs_collections = "market"
                            OR
                            if net_profit_score > 500
                             and frustration_score > 8
                            and cust_rec.last_sent_bonus_check < now - 60
                             and marketing_vs_collections = "market"


   Type:              foreign

   Side-effect:             yes

         side-effect-function:

               issue_and_send_check($50,cust_rec.name,cust_rec.address)
```

09/251998

09/251998

FIG. 23

```
Module:  call_priority

   Submodule of:  routing_decisions

   Input attributes:     business_value_of_call
                         frustration_score

   Output attributes:    call_priority

   Enabling condition:   true

   Type:            decision

   Computation:
      Rules:             if business_value_of_call < 25 then
                               call_priority <- 1

                         if 25 =< business_value_of_call < 100 then
                               call_priority <- 2

                         if 100 =< business_value_of_call < 500 then
                               call_priority <- 3

                         if 500 =< business_value_of_call then
                               call_priority <- 4

                         if frustration_score > 8 then
                               call_priority <- 4

                         if 6 =< frustration_score <= 8 then
                               call_priority <- 3

      Combining policy: call_priority_cp // high value wins with
                                            default result 2

   Side-effect:          no
```

FIG. 24

```
Module: calculate_skill
   Submodule of:   routing_decisions
   Input attributes:           business_value_of_call
                               marketing_vs_collections
                               IVR_choice
                               DNIS
                               web_destinations
   Output attributes:          skill
   Enabling condition:         true
   Type:           decision
   Computation:
      Rules:       if marketing_vs_collections = "collections"
                      then skill <- ["collections", infinity]

                   if business_value_of_call > 100
                      then skill <- ["high_tier", 40]

                   if DNIS = "australia_promotion" then
                      skill <- ["australia_promo", infinity]

                   if "Australia" in web_destinations[i].regions
                      for some i where web_destinations[i].date_last_modified >
                      now - 30 then
                      skill <- ["australia_promo", 20]

                   if cust_rec.estimated_income_bracket = ">100K-150K" then
                      skill <- ["australia_promo", 25]

                   if cust_rec.estimated_income_bracket = ">150K" then
                      skill <- ["australia_promo", 35]

                   if IVR_choice = "dom" then skill <- ["norm_tier_dom",30]

                   if IVR_choice = "intl" then skill <- ["norm_tier_intl",30]

                   if "US" in web_destinations[i].regions for some
                      i where web_destinations[i].date_last_modified >
                      now - 30 then
                      skill <- ["norm_tier_dom", 20]

                   if "US" not in web_destinations[i].regions for
                      some i where web_destinations[i].date_last_modified > now -
                      30 then
                      skill <- ["norm_tier_intl", 20]
      Combining policy:  skill_cp //weighted sum policy, and ties are
                            broken by ordering "collections",
                            "australia_promo", "high_tier",
                            "low_tier_intl", "low_tier_dom",
                            default is ⊥

   Side-effect:  no
```

09/251998

FIG. 25

```
Module:  calculate_on_queue_promo

    Submodule of:  routing_decisions

    Input attributes:        promo_hit_list

    Output attributes:       on_queue_promo

    Enabling condition:      DISABLE if business_value > 100 or

frustration_score > 5

    Type:            decision

    Computation:
        Rules:               if 60 < ACD.expected_wait_time(skill)
                                 then on_queue_promo <-
                                 promo_hit_list[#1]

                             if business_value_of_call < 30
                                 then on_queue_promo <- promo_hit_list[#1]

        Combining policy:    on-queue-promo-cp // first true wins, default
                                                  is 0

    Side-effect:             no
```

09/251998

09/251998

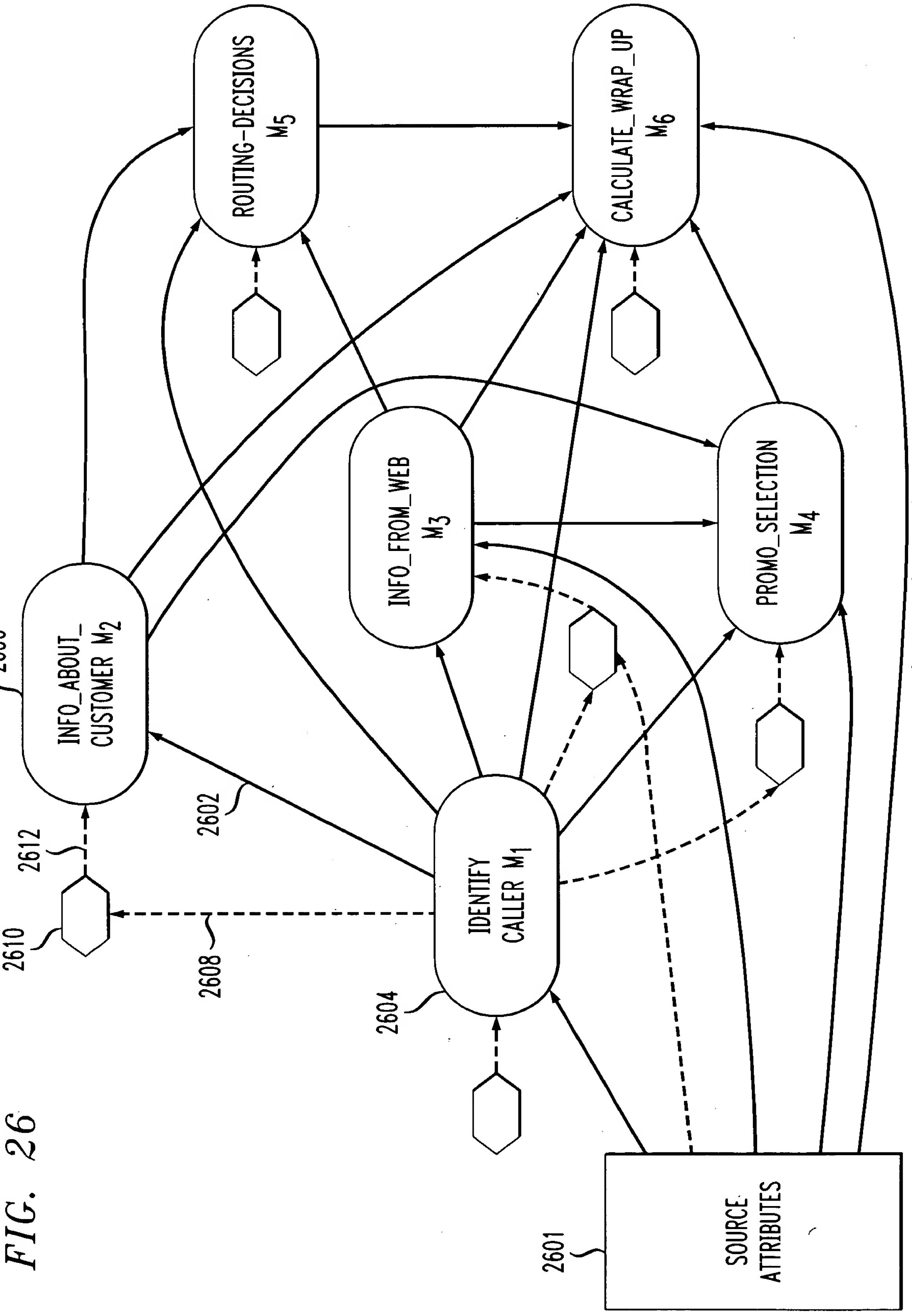


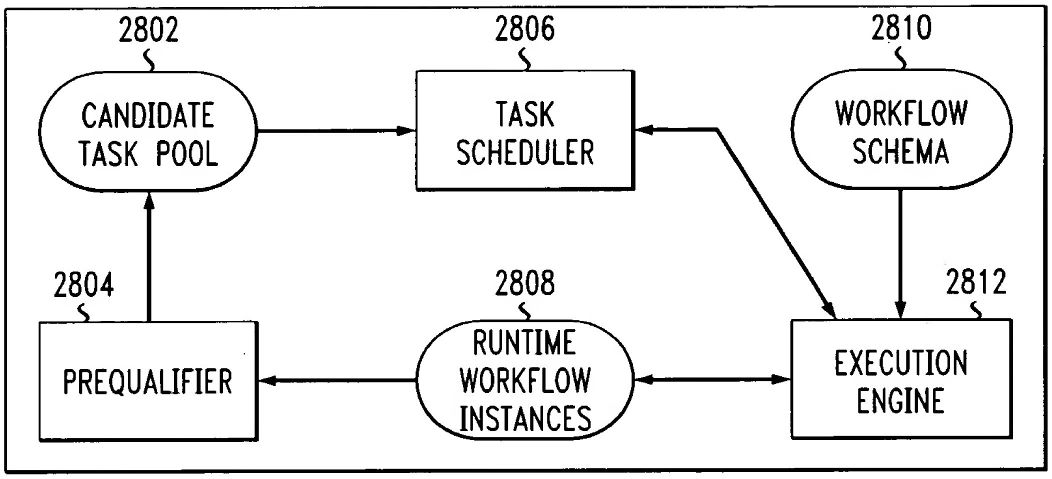
FIG. 26

FIG. 27

$\dfrac{\sigma \vdash e{:}t}{\sigma \vdash \text{value}(e){:}\ bool}$	VALUE
$\dfrac{\sigma \vdash f{:}AM{:}t_1 \text{x} \ldots \text{x} t_n \rightarrow t, \sigma \vdash e_1{:}t_1, \ldots \sigma \vdash e_n{:}t_n}{\sigma \vdash \text{Apply}(<f,e_1,\ldots,e_n>){:}t}$	APPLY
$\dfrac{\sigma \vdash e_1{:}t_1, \ldots \sigma \vdash e_n{:}t_n}{\sigma \vdash <e_1,\ldots,e_n>{:}<a_1{:}t_1,\ldots,a_n{:}t_n>}$	TUPLING
$\dfrac{\sigma \vdash e_1{:}t, \ldots, \sigma \vdash e_n{:}t}{\sigma \vdash \{e_1,\ldots,e_n\}{:}\{t\}}$	BAGGING
$\dfrac{\sigma \vdash e_1{:}t, \ldots, \sigma \vdash e_n{:}t}{\sigma \vdash [e_1,\ldots,e_n]{:}[t]}$	LISTING
$\dfrac{\sigma \vdash e{:}\{t\}}{\sigma \vdash \text{unitval}(e){:}t}$	UNITVAL
$\dfrac{\sigma \vdash <a_1{:}t_1,\ldots,a_n{:}t_n>}{\sigma \vdash e.a_i{:}t_i}$	PROJECTION ON TUPLES
$\dfrac{\sigma \vdash e{:}[t]}{\sigma \vdash e\#i{:}t}$	PROJECTION ON LISTS
$\dfrac{\sigma \vdash e_1{:}[t_1], \sigma \vdash e_2{:}t_2}{\sigma \vdash \text{factor}(e_1,e_2){:}[<\text{f\_}a{:}t_1, s\_a{:}t_2>]}$	FACTOR (ON LISTS)
$\dfrac{\sigma \vdash e_1{:}\{t_1\}, \sigma \vdash e_2{:}t_2}{\sigma \vdash \text{factor}(e_1,e_2){:}\{<\text{f\_}a{:}t_1, s\_a{:}t_2>\}}$	FACTOR (ON BAGS)
$\dfrac{\sigma \vdash f{:}t_1 \rightarrow t, \sigma \vdash S{:}[t_1]}{\sigma \vdash \text{map}(f)(S){:}[t]}$	MAP (ON LISTS)
$\dfrac{\sigma \vdash f{:}t_1 \rightarrow t, \sigma \vdash S{:}\{t_1\}}{\sigma \vdash \text{map}(f)(S){:}\{t\}}$	MAP (ON BAGS)
$\dfrac{\sigma \vdash id_\theta{:}t, \sigma \vdash \theta{:}t \text{x} t \rightarrow t, \sigma \vdash S{:}\{t\}}{\sigma \vdash \text{collect}\ (id_\theta, \theta)\ (S){:}t}$	COLLECT (ON BAGS)
$\dfrac{\sigma \vdash id_\theta{:}t, \sigma \vdash \theta{:}t \text{x} t \rightarrow t, \sigma \vdash S{:}[t]}{\sigma \vdash \text{collect}\ (id_\theta, \theta)\ (S){:}t}$	COLLECT (ON LISTS)

09/251998

FIG. 28



09/251998

FIG. 29

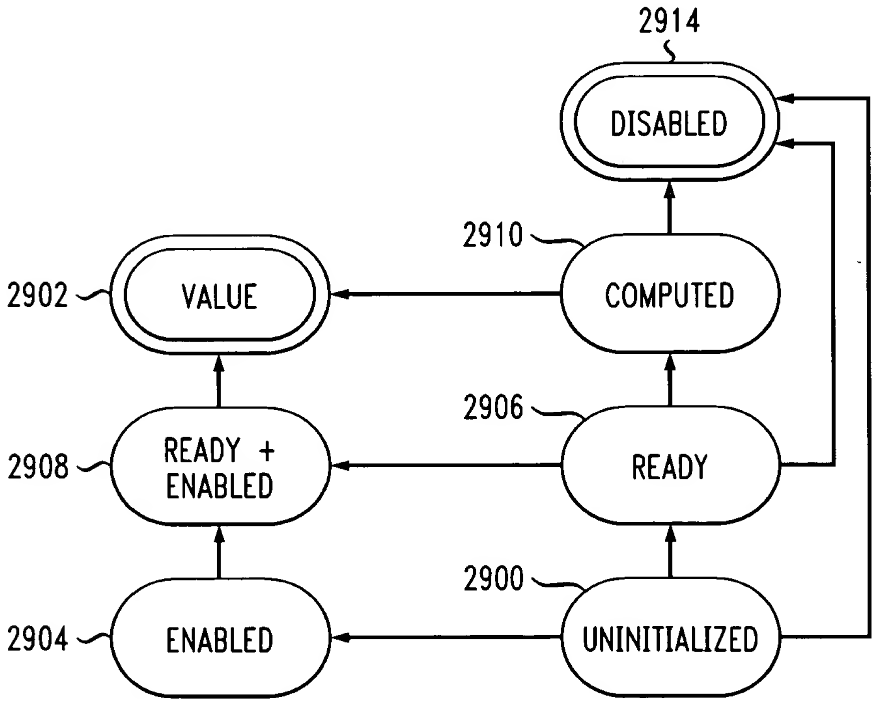
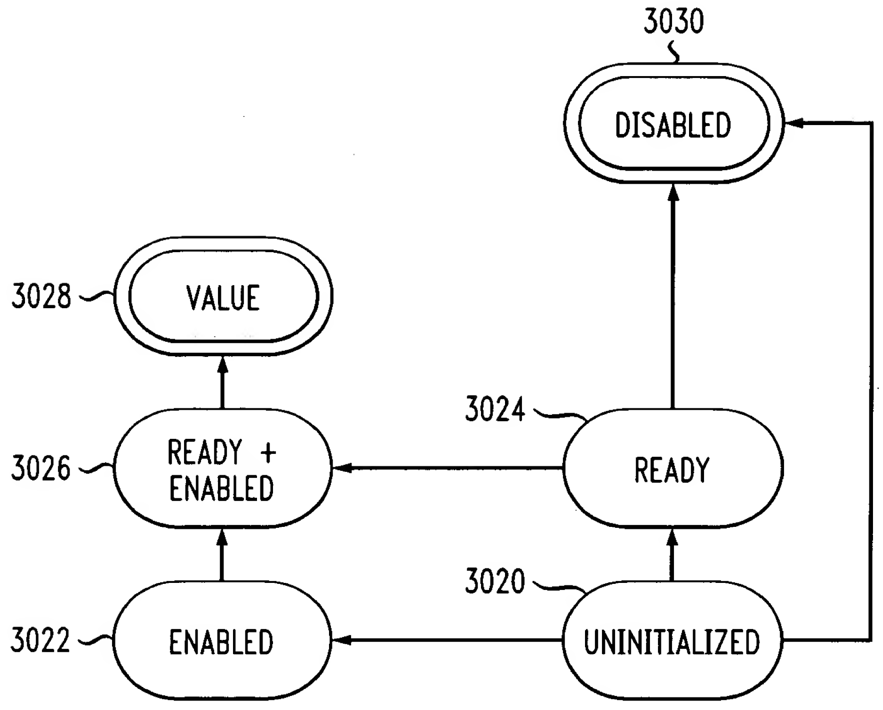
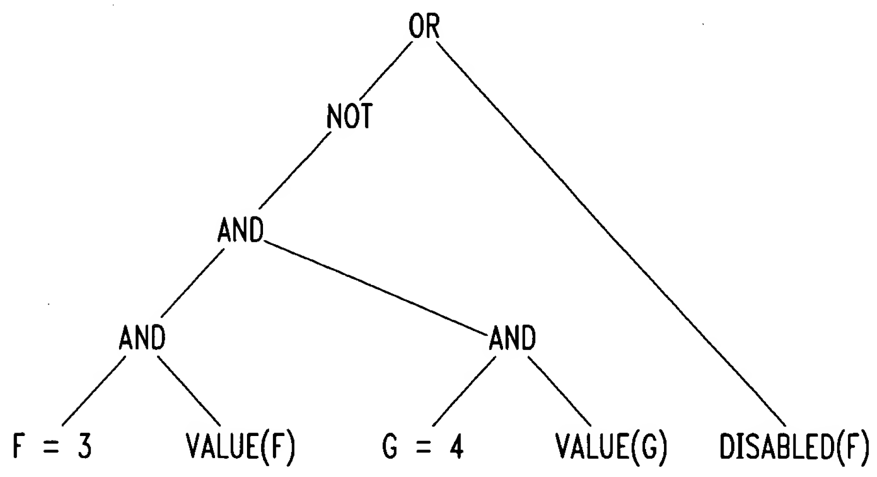


FIG. 30



09/251998

FIG. 31



09/251998

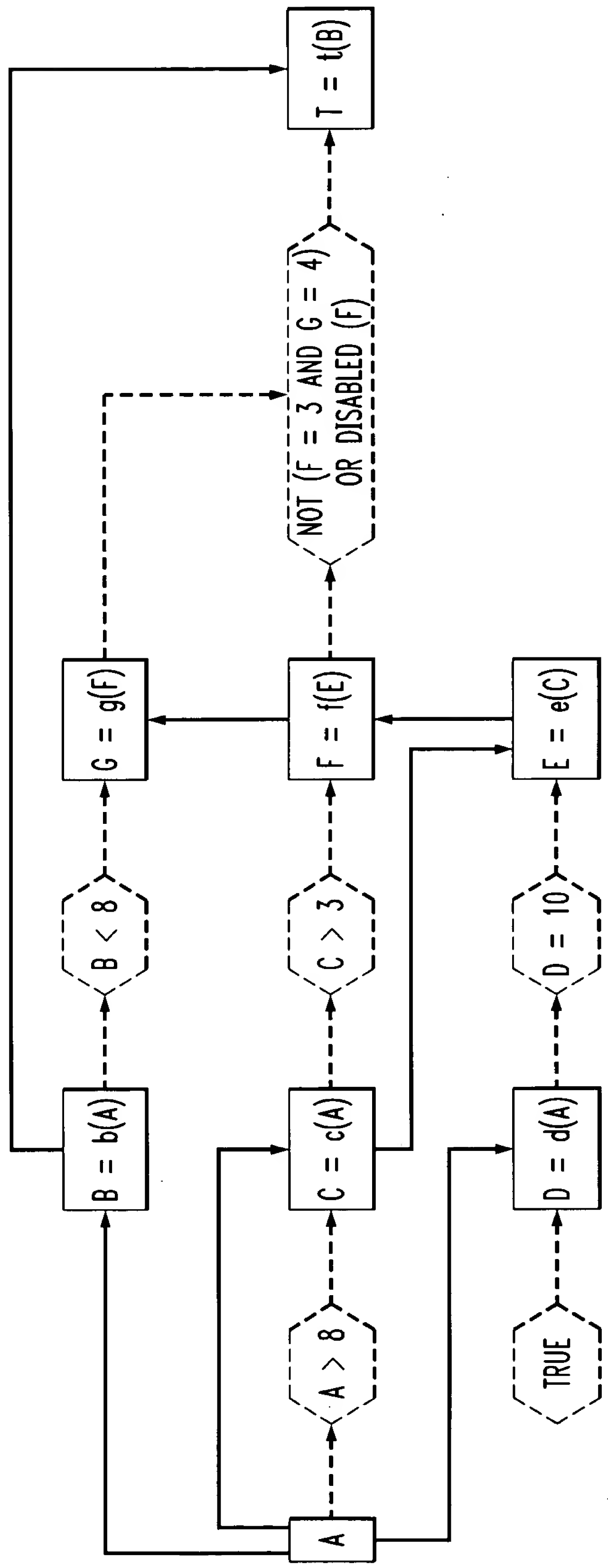
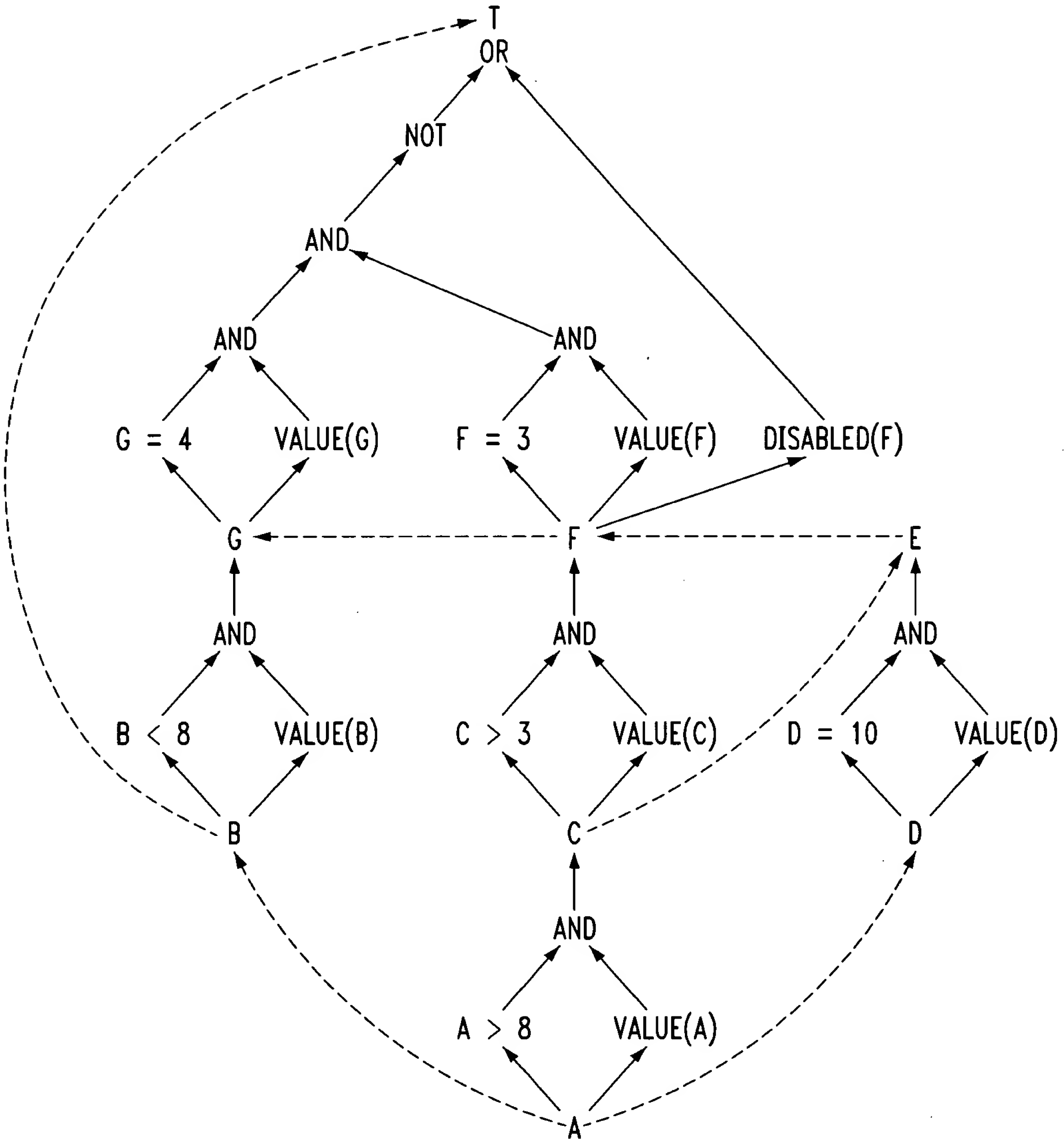


FIG. 32

FIG. 33


T
OR
NOT
AND
AND
AND
G = 4
VALUE(G)
F = 3
VALUE(F)
DISABLED(F)
G
F
E
AND
AND
AND
B < 8
VALUE(B)
C > 3
VALUE(C)
D = 10
VALUE(D)
B
C
D
AND
A > 8
VALUE(A)
A

09/251998

FIG. 34A

Global variables:

These variables are global to the whole execution of workflow instance 3402

- G: a dependency graph
- S: set of source attribute nodes of G
- T: set of target attribute nodes of G
- σ []: array of attribute states
- μ []: array of attribute values
- α []: array of three valued logic values (*true, false unknown*)
- *HIDDEN_EDGE*: set of hidden edges of G.
- *HIDDEN_ATT*: set of hidden attribute nodes of G.

Notations: 3404

- σ [A]: element of array σ[] that corresponds to the attribute node A in G
- μ [A]: element of array μ[] that corresponds to the attribute node A in G
- α [p]: element of array α[] that corresponds to the condition node p in G

09/251998

FIG. 34A (cont)

```
Initialization phase:                                                          3406
    procedure Init:
    Input:
        g: a dependency graph:
        So: source nodes in g
        Te: terminal nodes in g
    body:
    BEGIN init
        G:=g ; S: = So: T: = Te;
        /*Initialization of the states and values of attributes nodes */        3408
        FOR all the attribute nodes A in G DO
            IF A ∈ S  /* A is a source node */
                THEN σ[A] := READY + ENABLED
                ELSE σ[A] := UNITIALIZED;
        μ[A] :=NULL;
        END FOR
        /* Initialization of α-values of condition nodes */                      3410
        FOR all the condition nodes p in G DO
          α[A] := unknown;
        END FOR
        */ Initialization of the set of hidden edges and hidden nodes */          3412
        HIDDEN_EDGE:=∅; HIDDEN_ATT:=∅
    END init
```

09/251998

FIG. 34B

```
Increment                                                      3414
    Input:
        A : an attribute in G.                     3416
        v : a value for A.
    body:
        BEGIN increment
    μ [A] : = v;                       3418
    IF σ [A] = READY
        THEN propagate_att_change(A, COMPUTED)                 3420
    IF σ [A] = READY+ENABLED
        THEN propagate_att_change(A, VALUE)
END Increment

propagate_att_change                                                        3422
    Input:
        B : an attribute in G.                     3424
        σ : a state for B
    body:
    /* Set state for B*1
    IF ((σ[B] = ENABLED) AND ( σ = READY)) OR (σ[B] = READY) AND (σ =ENABLED))
        THEN σ [B] := READY+ENABLED                                3426
        ELSE σ [B] := σ;
```

09/251998

FIG. 34B
(cont)

3422
3428

```
/* push relevant information to the affected successor nodes */                                          3430
CASE : $\sigma$ [$B$] $\in$ {VALUE, COMPUTED} /* The value of $B$ is computed */
  /* try to evaluate predicate nodes that are using the value of $B$ */
  FOR each condition node $p$ of the form $pred(t_1, \cdots, t_n)$ such that $(B,p) \in G$ DO        3432
        IF $(B,p) \notin$ HIDDEN_EDGE        3434
              THEN
                  Hide_edge ($(B,p)$);        3436
                  IF $Eval(p) \neq unknown$ THEN $\alpha$ [$p$] :=$Eval(p)$; propagate_cond_change(p);        3438
  END FOR
/* check if the attributes nodes that have B as input parameters are READY */
FOR each attribute node C such that $(B, C) \in G$ DO
IF $\sigma$ [$B$] =VALUE THEN
        IF $(B, C) \notin HIDDEN\_EDGE$
              THEN
                  Hide_edge($(B,C)$);        3440
                  IF there exists no attribute node $D$ such that $(D, C) \notin HIDDEN\_EDGE$
                        THEN propagate_att_change ($C$ READY);
END FOR
CASE : $\sigma$ [$B$] = ENABLED
/* evaluates condition nodes of the form VALUE ($B$) and DISABLED ($B$) */
FOR each condition node $p$ of the form VALUE ($B$) or DISABLED ($B$) such that $(B,p) \in G$ DO
        IF $(B,p) \in HIDDEN\_EDGE$        3442
              THEN
                  Hide_edge($(B,p)$);
                  IF $p$ is of the form VALUE ($A$) THEN $\alpha[p]$ := $true$ ELSE $\alpha[p]$: = $false$;
                  propagate_cond_change($p$);
```

09/251998

FIG. 34C

```
   END FOR
CASE: $\sigma[B]$ = DISABLED
  /* evaluate condition nodes of the form VALUE ($B$) and DISABLED ($B$) */
  FOR each condition node $p$ of the form VALUE ($B$) or DISABLED ($B$) such that $(B,p) \in G$ DO     3444

    IF $(B,p) \notin$ HIDDEN_EDGE
       THEN
         Hide_edge (($B,p$));
         IF $p$ is of the form VALUE ($A$) THEN $\alpha[p]$ := false ELSE $\alpha[p]$ := true;
         propagate_cond_change($p$);
  END FOR
  /* check if the attribute nodes that have $B$ as input parameters are READY */
  FOR each attribute node $C$ such that $(B,C) \in G$ DO

    IF $(B,C) \notin$ HIDDEN_EDGE                                                                        3446
       THEN
         Hide_edge(($B,C$));
         IF there are no more attribute nodes $D$ such that $(D,C) \notin$ HIDDEN_EDGE
            THEN propagate_att_change ($C$,READY);
  END FOR                                                                                                3428
  /* If the attribute is stable then hide the attribute */                                               3422
    IF ($\sigma[B] \in$ {DISABLED, VALUE}) THEN Hide_node($B$);  3448
  END propagate_att_change
```

09/251998

FIG. 34C (cont)

```
propagate_cond_change                                                      3450
  Input:
    p: a condition node in G.
  body:
  BEGIN propagate_cond_change
  let n be the successor of p in G                                         3452

  IF $(p,n) \notin HIDDEN\_EDGE$
    THEN                                                                   3454
      Hide_edge ((p,n));                                                   3456
      CASE: n is OR condition node                                         3458
        IF ($\alpha[p]$ = true) THEN $\alpha[n]$: = true; propagate_cond_change(n); END IF;      3460
        IF $\alpha[p]$ = false AND for each condition node p' where $(p',n) \in G$, $(p',n) \in$  3462
        HIDDEN_EDGE
            THEN $\alpha[n]$ : = false; propagate_cond_change(n);END IF;
      CASE: n is a AND node                                                3464
        IF ($\alpha[p]$ = false) THEN $\alpha[n]$ :=false; propagate_cond_change(n);END IF;      3466
        IF $\alpha[p]$ = TRUE AND for each condition node p' where $(p',n) \in G$, $(p',n) \in$   3468
        HIDDEN_EDGE
            THEN $\alpha[n]$ : =TRUE; propagate_cond_change(n);END IF;
      CASE: n is NOT node                                                  3470
       $\alpha[n]$: =$\neg(\alpha[p])$ ; propagate_cond_change(n);
      CASE: n is an attribute node                                         3472
        IF ($\alpha[p]$ = true)
            THEN propagate_att_change(n,ENABLED)
            ELSE propagate_att_change(n,DISABLED);
    END propagate_cond_change
```

09/251998

FIG. 34D

3474

```
Hide_edge
  Input
      (n,n') : an edge in G.
body
BEGIN Hide_edge
HIDDEN_EDGE: = HIDDEN_EDGE U {(n,n')];

IF (there are no more edges (n,n'') ∈ G such that (n,n'') ∉ HIDDEN_EDGE
      THEN Hide_node(n)
END Hide_edge
```

3476

```
Hide_node
    Input
        n : a node in g.
    body
    BEGIN Hide_node
    HIDDEN_ATT: = HIDDEN_ATT U {n}

    FOR each edge (n',n) ∈ g such that (n',n) ∉ HIDDEN_EDGE) DO
        Hide_edge (n',n)
    END FOR
END Hide_node
```

09/251998

FIG. 35A

3502

Global variables:

These variables are global to the whole execution of workflow instance

3504

```
G : a dependency graph
S : set of attribute nodes of G /* this set contains the source nodes */
T : set of attribute nodes of G /* this set contains target nodes */
σ  [] : array of attribute states
α  [] : array of three valued logic values (true, false unknown)
HIDDEN_EDGE : set of edges of G.
HIDDEN_ATT : set of attribute nodes of G.

T_N[][] : Matrix of integers that associates an integer value to each pair (p,A) where p is a
condition node and A is an attribute node
  in G
 /* T_N[p][A] = 0 means that the attribute A is True_necessary for the condition node
 p*/

F_N[][] : Matrix of integers that associates an integer value to each pair (p,A) where p is a
   condition node and A is an attribute node in G
 /*F_N[p][A] = 0 means that the attribute A is False_necessary for the condition node p*/

V_N [][] : Matrix of integers associates an integer value to each pair (B,A) where B and A
   are attribute nodes in G
 /*V_N[B][A] = 0 means that the attribute A is Value_necessary for the attribute note B*/
```

09/251998

FIG. 35A
(cont.)

```
S_N[][] : Matrix of integers associates an integer value to each pair (B,A) where
   B and A are attribute nodes in G
 /*S_N[B][A] = 0 means that the attribute A is Stable_necessary for the attribute node B */

N[] : Array of boolean
   N[A] = true means that the attribute A is computed as necessary/*
   N[A] = false means that the attribute A is not computed as necessary*/
```
3502

Notations:

nb_pred(p) : number of predecessors of p in G

```
Initialization phase:
  procedure Init :
  Input:
    g : a dependency graph:
    So : source nodes in g
    Te : terminal nodes in g
  body:
  BEGIN N_init
```
3506

09/251998

FIG. 35B

3506

3510

```
Init() 3508

/* Initialization of T_N,F_N,S_N,V_N */
FOR all the condition nodes p in G DO
  FOR all the attribute nodes A in G DO

    CASE : p is an OR node:
      T_N[p][A] := nb_pred(p):                /* rule 1 */   3511
      F_N[p][A] := 1;                         /* rule 2 */

    CASE : p is an AND node:
      T_N[p][A] := 1;                         /* rule 3 */
      F_N[p][A] := nb_pred(p);                /* rule 4 */

    CASE : p is a NOT node:
      T_N[p][A] := 1;                         /* rule 5 */
      F_N[p][A] := 1;                         /* rule 6 */

    CASE : p is a node of the form VAL(B) or DIS(B):
      T_N[p][A] := 1;                         /* rules 7 and 9 */
      F_N[p][A] := 1;                         /* rules 8 and 10 */

    CASE: p is a node of the form pred(t_1,...t_n):
      T_N[p][A] := 1;                         /* rule 11 */
      F_N[p][A] := 1;                         /* rule 12 */

  END FOR
END FOR
```

09/251,998

FIG. 35B
(cont.)

3506

```
FOR all the attributes nodes $A$ in $G$ DO                      3512
  FOR all the attribute nodes $B$ in $G$ DO
    $S\_N[A][B] := 1$; $V\_N[A][B] := 1$
  END FOR
END FOR

FOR all the attributes nodes $A$ in $G$ DO                      3513
  $N[A] := false$
END FOR

END N_init
```

3526

```
N-Increment
Input:
  $A$ : an attribute in $G$.
  $v$ : a value for $A$.
Variables/* Global to one execution of the increment phase (for one execution step) */      3518

prev_E: set of attribute nodes
      /* used to store the nodes that were READY+ENABLED or ENABLED (in a
      previous execution of N-increment) */

prev_HIDDEN_EDGE :/* set of edges*/
      used to store the edges that were previously hidden (in the previous steps) */
```

09/251998

FIG. 35C

```
prev_T_N: set of pairs (p,A) where p is a condition node and A is an attribute node
    /* used to denote the elements of T_N[p][A] that were set to 0 in a previous
    execution of N-increment*/

prev_F_N: set of pairs (p,A) where p is a condition node and A is an attribute node
    /* used to denote the elements of F_N[p][A] that were set to 0 in a previous
    execution of N-increment*/
```
3518

```
Δ_E: set of attribute nodes
  /* used to store the new ENABLED or READY+ENABLED attribute nodes that were
  neither ENABLED nor READY+ENABLED in the previous steps. */
Δ_HIDDEN_EDGE: set of edges
  /* used to store the new hidden edges */
```
3520

```
new_V_N : set of pair (A,A) where A is an attribute node
  /* used to store the positions of elements of V_N[][] whose new value is zero due to
  case 1 */
new_S_N : set of pair (B,A) where B and A are attribute nodes
  /* used to store the positions of elements of S_N[][] whose new value is zero due to
  case 2 */
new_T_N : set of pair (p,A) where p is a predicate node and A is an attribute node
  /* used to store the positions of elements of T_N[][] whose new value is zero due to
  some new hidden edges (case 3) */
new_F_N : set of pair (p,A) where p is a predicate node and A is an attribute node
  /* used to store the positions of elements of F_N[][] whose new value is zero due to
  some new hidden edges (case 4) */
```
3522

3526

09/251998

FIG. 35D

```
body:
BEGIN N_increment

/* preparation step: */                                                   3524
prev_HIDDEN_EDGE := HIDDEN_EDGE;
prev_E := {A|A is an attribute node in G and σ [A] ∈ {READY+ENABLED,
ENABLED}

Increment(A,v)                                                            3526

/* Instigation step : Compute new necessary properties according to the instigation   3528
cases*/

Case 1:                                                                   3530
  Δ_E := {A|A is an attribute node in G and σ [A] ∈ {READY+ENABLED, ENABLED}
  and A ∉ prev_E}
  new_V_N := ∅;
  FOR each attribute node A in Δ_E DO
    V_N[A][A] := 0; new_V_N := new_V_N U {(A,A)}/* a node is value_necessary for
    itself*/
  END FOR
```

09/251998

FIG. 35D
(cont.)

3528

```
Case 2:                                                                          3532
 $new\_S\_N := \emptyset$;
 FOR each attribute node $B$ in $\Delta\_E$ DO
  FOR each attribute node in $A$ in $G$ such that $\sigma[A] \in$ {READY+ENABLED,
  ENABLED} DO
   IF $V\_N[B][A] = 0$ and $S\_N[B][A] = 1$
      THEN $S\_N[B][A] = 0$; $new\_S\_N := new\_S\_N \cup \{(B,A)\}$ /*    rule (13)*/
   END FOR
  END FOR

$\Delta\_HIDDEN\_EDGE := HIDDEN\_EDGE - prev\_HIDDEN\_EDGE$                          3534
$prev\_T\_N := \{(p,A) \mid T\_N[p][A] = 0\}$
$prev\_F\_N := \{(p,A) \mid F\_N[p][A] = 0\}$
$new\_T\_N := \emptyset$;
$new\_F\_N := \emptyset$;

FOR all edges $(n,p) \in \Delta\_HIDDEN\_EDGE$ such that $p \notin HIDDEN\_ATT$ and $p$ is a
condition node DO
 FOR all attribute nodes $A$ such that $\sigma(A) \notin$ {COMPUTED, VALUE, DISABLED}
 DO
  CASE: 3                                                                          3536
 CASE : $p$ is an OR node:
  IF $(n,A) \notin prev\_T\_N$
  THEN
    $T\_N[p][A] := T\_N[p][A] - 1$;            /* rule (1)*/
    IF $T\_N[p][A] = 0$ THEN $new\_T\_N := new\_T\_N \cup \{(p,A)\}$
```

09/251998

FIG. 35E

```
CASE: 4
CASE : $p$ is an AND node :
IF (n,A) $\not\in$ prev_F_N /* same reasoning as for OR nodes but with rule 4*/
THEN
  F_N[p][A] := F_N[p][A] -1;            /* rule (4)*/
  IF F_N[p][A] = 0 THEN new_F_N := new_F_N U {(p,A)}
 END FOR
END FOR
```

3538

3528

09/251998

FIG. 35E (cont.)

```
/* Propagation step */
   New_propagate(new_V_N, new_S_N, new_T_N, new_F_N)        3540
END N_Increment

New_propagate
  Input:
   new_V_N : set of pairs (A,A) where A is an attribute node
   new_S_N : set of pairs (B,A) where B and A are attribute nodes
   new_T_N : set of pairs (p,A) where p is a condition node in G and A is an attribute
   node
   new_F_N : set of pairs (p,A) where p is a condition node in G and A is an attribute
   node
  body:
  FOR each pair (A,A) in new_V_N DO
    propagate_V_N(A,A)
    FOR each attribute node B such that $(A,B) \in G$ and $(A,B) \notin$ HIDDEN_EDGE
      V_N[B][A] := 0; propagate_V_N(B,A)/*      rule (16) */        3546
    END FOR
  END FOR
  FOR each pair (B,A) in new_S_N DO
    propagate_S_N(B,A)
  END FOR
   FOR each pair (p,A) in new_T_N DO
     propagate_T_N(p,A)
  END FOR
  FOR each pair (p,A) in new_F_N DO
     Propagate_F_N(p,A)
  END FOR
END N-propagate
```

3544

3542

09/251998

OIPE JC166
JAN 2 7 2003
PATENT & TRADEMARK OFFICE

FIG. 35F

3542

3548
```
propagate_V_N
  Input :
    B : an attribute node in G.
    A : an attribute node in G./* A in newly Value_necessary for B */
  body:
  IF σ [B] = ENABLED and S_N[B][A] = 1
     THEN S_N[B][A] = 0; propagate_S_N(B,A)          /*rule (13)*/   3550
ELSE let p be the condition node such that (p,B) ∈ G.
      IF F_N[p][A]=0 and S_N[B][A] = 1
        THEN S_N[B][A] = 0; propagate_S_N(B,A)         /*rule (14)*/  3552
END IF
FOR each condition node p of the form pred(t_1...,t_n)
      such that (B,p) ∈ g and (B,p) ∉ HIDDEN_EDGE DO
IF T_N[p][A] = 1
   THEN T_N[p][A] : = 0;propagate_T_N(p,A)           /*rule (11)*/   3554
IF F_N[p][A] = 1
  THEN F_N[p][A] : =0;propagate_F_N(p,A)             /*rule (12)*/   3556
 END FOR
END propagate_V_N
```

3558
```
propagate_S_N
  Input:
    B : an attribute node in G.
    A : an attribute node in G./* A is newly Stable_necessary for B*/
  body:
  FOR each attribute node C such that (B,C) ∈ g and (B,C) ∉ HIDDEN_EDGE DO
    IF V_N[C][A] = 1 THEN V_N[C][A] = 0;propagate_V_N(C,A) /* Rule 17 */   3560
  END FOR
  IF B ∈ T THEN N[A] : =true   3562
END propagate_S_N
```

09/251998

FIG. 35F
(cont.)

3564

3542

```
propagate_F_N
 Input:
  $p$ : a condition node in $G$.
  $A$ : an attribute node in $G$./* $A$ is newly False_necessary for $p$*/
 body:
  let $n$ be the successor of $p$ in $G$
  IF $(p,n) \in HIDDEN\_EDGE$
     THEN
      CASE : $n$ is an OR or AND node
        IF $F\_N[n][A] > 0$
           THEN
           $F\_N[n][A] := F\_N[n][A] - 1$;        /***rules (2) and (4)***/
           IF $F\_N[n][A] = 0$ THEN propagate $F\_N\ (n,A)$
        CASE : $n$ is a NOT node
         IF $T\_N[n][A] = 1$ THEN $T\_N[n][A]$ : =0;propagate_T_N($n,A$) /***rule (6)***/
        CASE : $n$ is an attribute node
         IF $(T\_N[p][A] = 0$ or $V\_N[n][A] = 0$ and $S\_N[n][A] = 1$
            THEN $S\_N[n][A] = 0$;propagate_S_N($n,A$)     /***rules (14) and (15)***/
           FOR each condition node $p'$ of the form VALUE $(n)$
              such that $(n,p') \in g$ and $(n,p') \notin HIDDEN\_EDGE$ DO
         IF $F\_N[p'][A] = 1$ THEN $F\_N[p'][A]$ : =0;propagate_F_N($p',A$) /***rule (8)***/
        END FOR
        FOR each condition node $p'$ of the form DISABLED $(n)$
             such that $(n,p') \in G$ AND $(n,p') \notin HIDDEN\_EDGE$ DO
      IF $T\_N[p'][A] = 1$ THEN $(T\_N[p'][A]$ :=0;propagate_T_N($p',A$) /***rule (10)***/
     END FOR
END propagate_F_N
```

09/251998

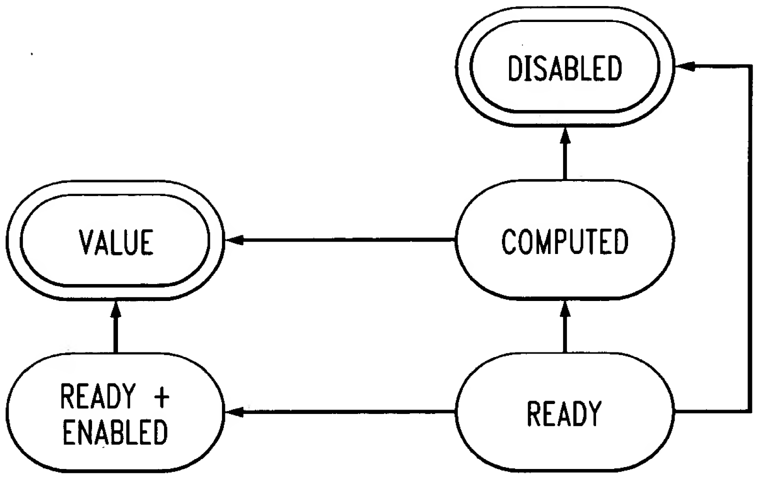
FIG. 35G

3542

3566

```
propagate_T_N
 Input:
  p : a condition node in G.
  A : an attribute node in G/* A is newly True_necessary for p */
 body:
  let n be the successor of p in G
  IF (p,n) ∉ HIDDEN_EDGE
     THEN
      CASE : n is an OR or AND node
       IF T_N[n][A]>0
          THEN
          T_N[n][A] := T_N[n][A] - 1; /*rules (1) and (3)*/
          IF T_N[n][A] = 0 THEN propagate_T_N(n,A)
      CASE : n is a NOT node
       IF F_N[n][A] = 1 THEN F_N[n][A] : = 0; propagate_F_N(n,A) /*rule (5) */
      CASE : n is an attribute node
       IF F_N[p][A] = 0 and S_N[n][A] = 1
          THEN S_N[n][A] =0;propagate_S_N(n,A) /*rule (15)*/
       FOR each condition node p' of the form VALUE(n)
            such that (n,p') ∈ G and (n,p') ∉ HIDDEN_EDGE DO
          IF T_N[n][A] = 1 THEN
                 T_N[p'][A] : = 0;propagate_T_N(p',A)     /*rule (8)*/
       END FOR
       FOR each condition node p' of the for DISABLED (n)
          Such that (n,p') ∈ G and (n,p') ∉ HIDDEN_EDGE DO
        IF F_N[n][A] = 1 THEN
        F_N[p'][A] : =0;propagate_F_N(p',A)      /*rule (9)*/
       END FOR
END propagate_T_N
```

FIG. 36



09/251998

FIG. 37

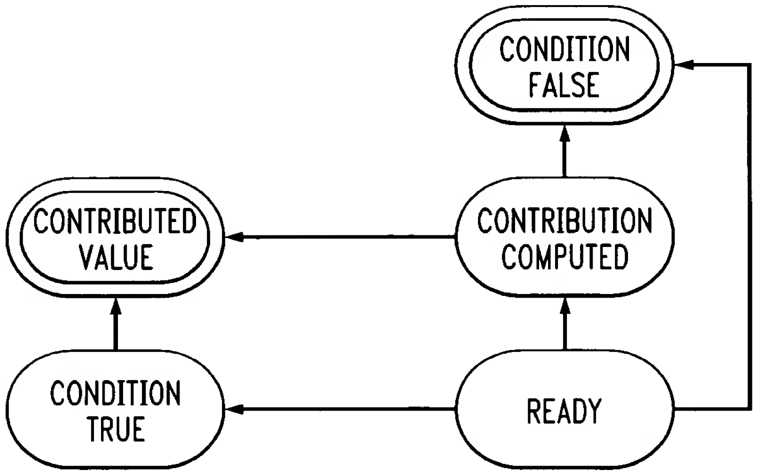
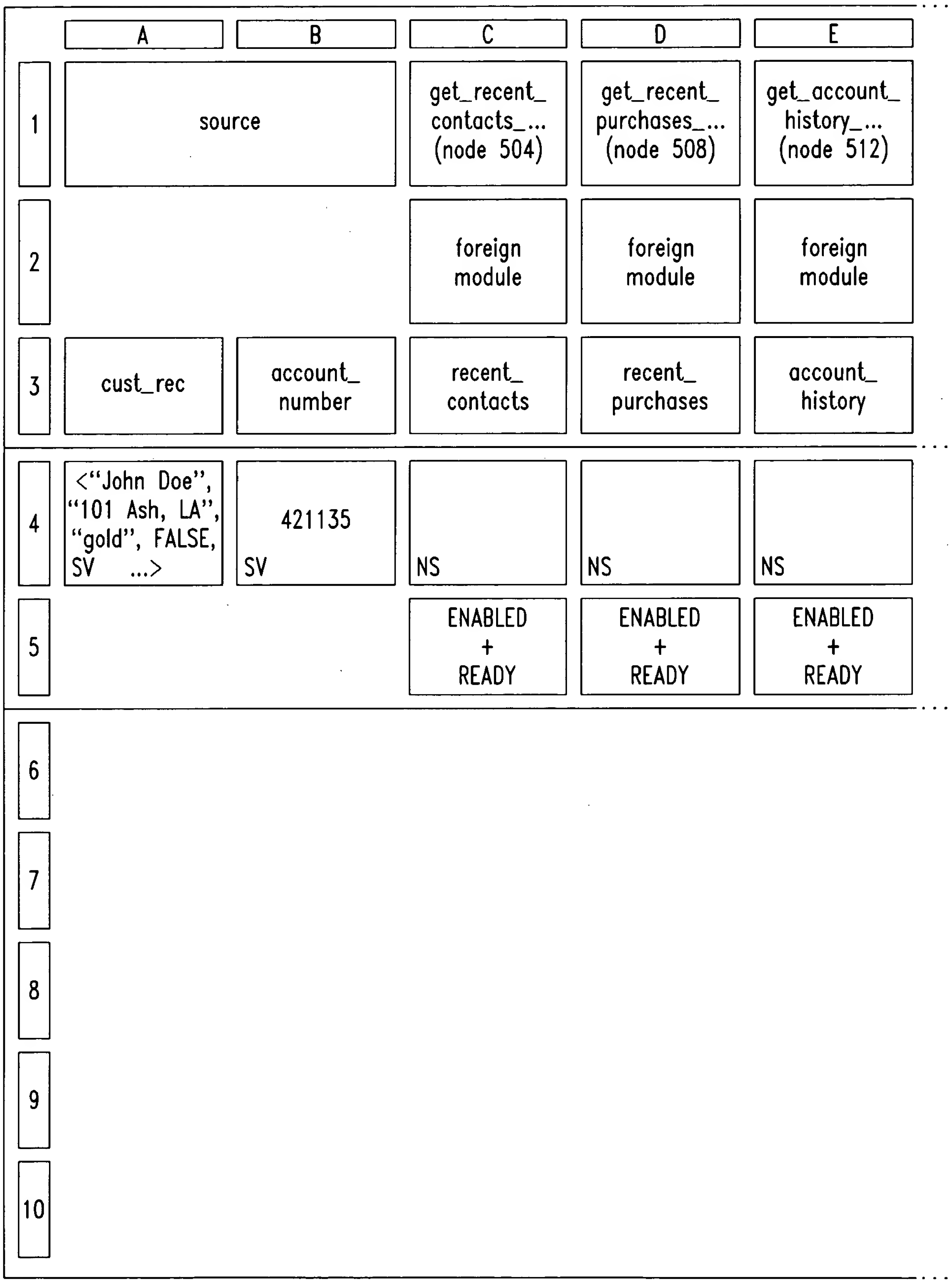


FIG. 38



	A	B	C	D	E
1	source		get_recent_ contacts_.... (node 504)	get_recent_ purchases_... (node 508)	get_account_ history_... (node 512)
2			foreign module	foreign module	foreign module
3	cust_rec	account_ number	recent_ contacts	recent_ purchases	account_ history
4	<"John Doe", "101 Ash, LA", "gold", FALSE, ...> SV	421135 SV	NS	NS	NS
5			ENABLED + READY	ENABLED + READY	ENABLED + READY
6					
7					
8					
9					
10					

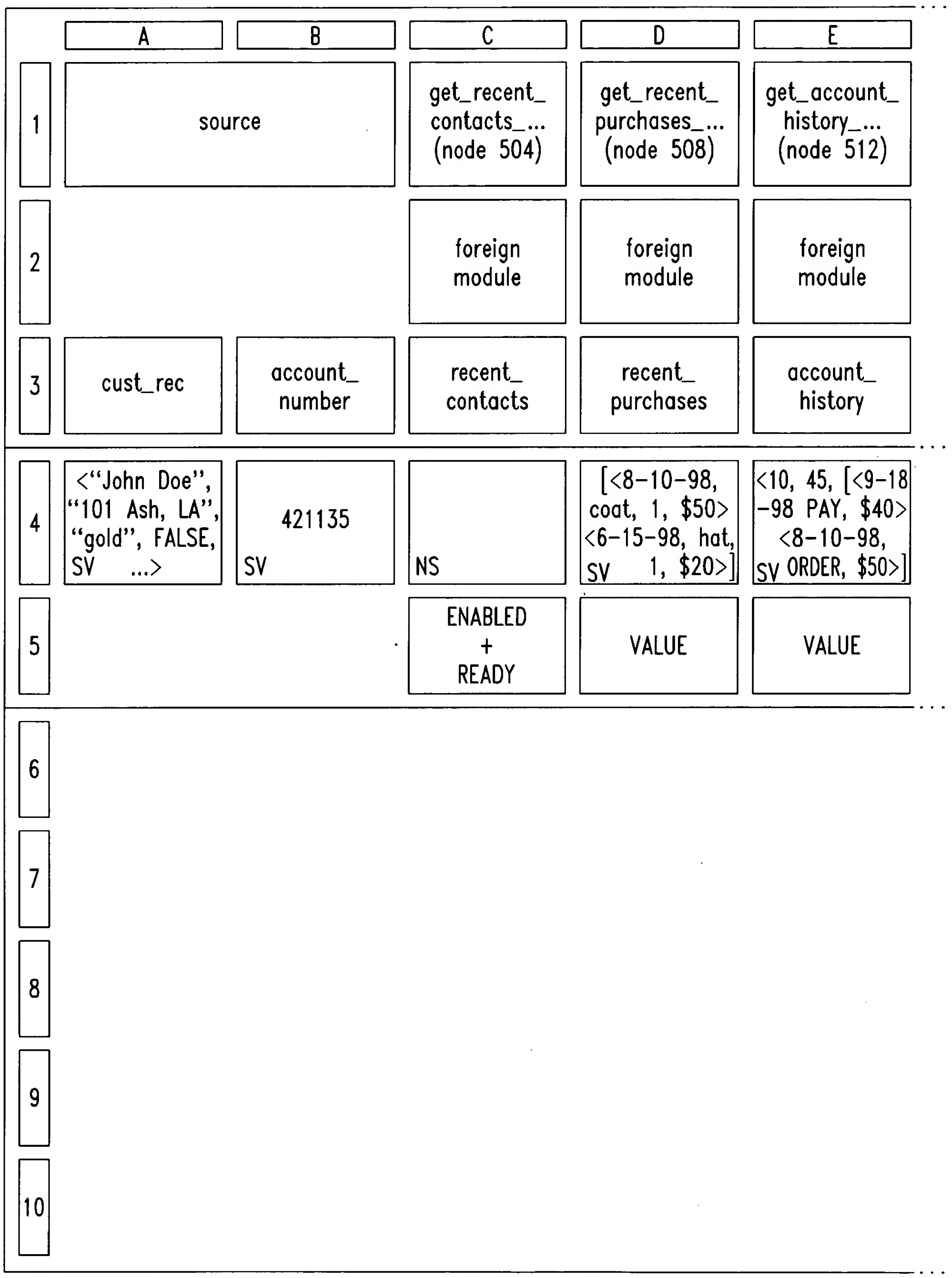
09/251998

FIG. 38 (cont)

F	G	H	I	J
calculate_ frustration_ score (node 516)	calculate_ net_profit_ score (node 520)	calculate_ late_payments_ score (node 524)	calculate cust_value (node 528)	calculate_ marketing_vs_ collections (node 532)
"add contribs. of true rules and round up, to max of 10"	"add contribs. of true rules"	"true rule wins; default is 0"	"add contribs. of true rules and round up, to max of 100"	"any true rule gives collect; default is marketing"
frustration_ score	net_profit_ score	late_ payment_ score	cust_ value	marketing_ vs_ collections
NS	NS	NS	NS	NS
READY	READY	READY	ENABLED + READY	READY
READY	READY	⊥	READY	"collect" C-C
READY	READY	condition true	⊥	
	READY	⊥	10 C-V	
	⊥		⊥	
	50 C-V		READY	

09/251998

FIG. 39



	A	B	C	D	E
1	source		get_recent_ contacts_... (node 504)	get_recent_ purchases_... (node 508)	get_account_ history_... (node 512)
2			foreign module	foreign module	foreign module
3	cust_rec	account_ number	recent_ contacts	recent_ purchases	account_ history
4	<"John Doe", "101 Ash, LA", "gold", FALSE, SV ...>	421135 SV	NS	[<8-10-98, coat, 1, $50> <6-15-98, hat, SV 1, $20>]	<10, 45, [<9-18 -98 PAY, $40> <8-10-98, SV ORDER, $50>]
5			ENABLED + READY	VALUE	VALUE
6					
7					
8					
9					
10					

09/251998

FIG. 39 (cont)

09/251998

F	G	H	I	J
calculate_ frustration_ score (node 516)	calculate_ net_profit_ score (node 520)	calculate_ late_payments_ score (node 524)	calculate cust_value (node 528)	calculate_ marketing_vs_ collections (node 532)
"add contribs. of true rules and round up, to max of 10"	"add contribs. of true rules"	"true rule wins; default is 0"	"add contribs. of true rules and round up, to max of 100"	"any true rule gives collect; default is marketing"
frustration_ score	net_profit_ score	late_ payment_ score	cust_ value	marketing_ vs_ collections
NS	⊥ SV	9 SV	NS	NS
READY	DISABLED	VALUE	ENABLED + READY	ENABLED + READY
READY	⊥	⊥	⊥	"collect" C-C
READY	READY	9 C-V	⊥	
	-9 C-V	⊥	10 C-V	
	⊥		⊥	
	50 C-V		READY	

FIG. 40A

```
Initialization
  Based on the DL specification, compute rows 1, 2, and 3 of the display;   4002
  For source attribute cells or row 4 do:                                     4004
      For each source attribute with value, insert value and apply
  "attribute_value_indication";
      For each source attribute that is disabled, apply
  "attribute_disabled_indication";
  For each non-decision module                                                4006
      In row 5, apply "module_uninitialized_indication";
      In row 4, apply "attribute_uninitialized_indication";
  For each decision module                                                    4008
      In row 5, apply "module_ready_indication";
      In row 4, apply "attribute_uninitialized_indication";
  For each cell in rows 6,7,8,.,apply "rule_ready_indication"                 4010

Iteration
  For each event of execution engine do
  Case on event_type
      non_dec_module_enabled:                                                 4012
          in row 5, apply "module_enabled_indication";

      non_dec_module_ready:                                                   4014
          in row 5, apply "module_ready_indication";

      non_dec_module_ready+enabled:                                           4016
          in row 5, apply "module_ready+enabled_indication";

      non_dec_module_computed:                                                4018
          in row 5, apply "module_computed_indication";
          in row 4, label corresponding attribute cell with the value computed
      and apply
                 "attribute_computed_indication";

      non_dec_module_value:                                                   4020
          in row 5, label cell for this module as "value" and apply
      "module_value_indication";
          in row 4, label corresponding attribute cell with value assigned and
      apply
                 "attribute_value_indication"
```

09/251998

FIG. 40B

```
non_dec_module_disabled:                                                    4022
    in row 5, label cell for this module as "disabled" and apply
        "module_disabled_indication";
    in row 4, label corresponding attribute cell with "⊥" and apply
        "attribute_disabled_indication"

dec_module_enabled+ready:                                                   4024
    in row 5, label cell with "enabled+ready" and apply
        "module_enabled+ready_indication";

dec_module_computed:                                                        4026
    in row 5, label cell with "computed" and apply
"module_computed_indication";
    in row 4, label cell with the computed value and apply
        "attribute_computed_indication";

dec_module_value:                                                           4028
    in row 5, label cell with "value" and apply
"module_value_indication";
    in row 4, label cell with the computed value and apply
"attribute_value_indication";

dec_module_disabled:                                                        4030
    in row 5, label cell with "disabled" and apply
    "module_disabled_indication";
    in row 4, label cell with "⊥" and apply
"attribute_disabled_indication";

comp_rule_condition_true:                                                   4032
    to corresponding cell, apply "rule_cond_true_indication";

comp_rule_contribution_computed:                                            4034
    to corresponding cell, label with computed value and apply
        "rule_contribution_computed_indication";

comp_rule_contributed_value:                                                4036
    to corresponding cell, label with computed value and apply
        "rule_contributed_value_indication";

comp_rule_condition_false:                                                  4038
    to corresponding cell, label with "⊥ " and apply
"rule_condition_false_indication";

EndCase
```

09/251998